IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
operating on the graphics floating point data in the graphics pipeline; and
storing the graphics floating point data to a buffer;
wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data;
wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

2.-3. (Cancelled)

4. (Previously Amended) The method as recited in claim 12, wherein the fragment data includes color data.

5. (Previously Amended) The method as recited in claim 12, wherein the fragment data includes depth data.

6. (Original) The method as recited in claim 1, wherein the graphics floating point data is only constrained by an underlying data type.

7. (Original) The method as recited in claim 1, wherein the buffer serves as a texture map.

8. (Currently Amended) A computer program product for buffering data produced by a computer graphics pipeline, comprising:

(a) computer code for producing graphics floating point data in a graphics pipeline;

(b) computer code for operating on the graphics floating point data in the graphics pipeline; and

(c) computer code for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data;

(e) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

9.-10. (Canceled)

11. (Previously Amended) The computer program product as recited in claim 8, wherein the fragment data includes color data.

12. (Previously Amended) The computer program product as recited in claim 8, wherein the fragment data includes depth data.

13. (Original) The computer program product as recited in claim 8, wherein the graphics floating point data is only constrained by an underlying data type.

14. (Original) The computer program product as recited in claim 8, wherein the buffer serves as a texture map.

15. (Currently Amended) A system for buffering data produced by a computer graphics pipeline, comprising:

(a) logic for producing graphics floating point data in a graphics pipeline;

(b) logic for operating on the graphics floating point data in the graphics pipeline; and

(c) logic for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data;

(e) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

16. (Currently Amended) A buffering apparatus, comprising:

(a) a buffer capable of storing graphics floating point data produced by a graphics pipeline;

(b) wherein the graphics floating point data includes fragment data received from a rasterizer that is stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data;

(c) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

17. (Currently Amended) A system for buffering data produced by a computer graphics pipeline, comprising:

(a) means for producing graphics floating point data in a graphics pipeline;

(b) means for operating on the graphics floating point data in the graphics pipeline;

and

(c) means for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data;

(e) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

18. (Currently Amended) A method for buffering data produced by a computer graphics pipeline, comprising:

(a) producing graphics floating point data in a graphics pipeline;

(b) operating on the graphics floating point data in the graphics pipeline; and

(c) storing the graphics floating point data to a buffer;

(d) wherein the buffer serves as a texture map;

(e) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

19. (Currently Amended) A buffering apparatus, comprising:

(a) a buffer capable of storing graphics floating point data produced by a graphics pipeline;

(b) wherein the buffer serves as a texture map;

(c) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

20. (Currently Amended) A method for buffering data during multi-pass rendering,

comprising:

(a) operating on graphics floating point data during a rendering pass in a graphics pipeline;

(b) reading the graphics floating point data from a buffer during the rendering pass;

(c) storing the graphics floating point data to the buffer during the rendering pass; and

(d) repeating (a) – (c) during additional rendering passes utilizing results of a previous rendering pass;

(e) wherein the graphics floating point data is packed in the graphics pipeline, the packing facilitating storage of at least two quantities in a single buffer in a single pass.

21. (Original) The method as recited in claim 20, wherein the operating includes deferred shading.

22. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
packing the graphics floating point data in the graphics pipeline; and
storing the graphics floating point data to a buffer;
wherein the packing facilitates storage of at least two quantities in a single buffer in a single pass.

23. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
unpacking the graphics floating point data in the graphics pipeline; and
operating on the unpacked graphics floating point data in the graphics pipeline;

wherein the unpacking facilitates storage of at least two quantities in a single buffer in a single pass.

24. (Original) A method for buffering data produced by a computer graphics pipeline, comprising:
operating on graphics floating point data in a graphics pipeline;
producing the graphics floating point data in the graphics pipeline;
determining whether the graphics pipeline is operating in a programmable mode utilizing a command associated with a graphics application program interface;
if it is determined that the graphics pipeline is not operating in the programmable mode, performing standard graphics application program interface operations on the graphics floating point data; and
if it is determined that the graphics pipeline is operating in the programmable mode:
storing the graphics floating point data to a frame buffer,
wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface extension for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

25. (Previously Presented) The buffering apparatus as recited in claim 19, wherein the buffer serves as the texture map by using previous rendering results via an extension of an application program interface.

26. (Previously Presented) The method as recited in claim 1, wherein the buffer includes a frame buffer.

27. (New) The method as recited in claim 1, wherein the packing converts "x" and "y" components of a single operand into a 16-bit floating-point format, packs a bit representation of the "x" and "y" components into a 32-bit value, and replicates the 32-bit value to each of four components of a result vector.

28. (New) The method as recited in claim 1, wherein the packing converts four components of a single operand into a plurality of 8-bit signed quantities, the 8-bit signed quantities being represented in a bit pattern where '0' bits correspond to -128/127 and '1' bits correspond to +127/127, where a bit representation of the converted four components are packed into a 32-bit value, the 32-bit value being replicated to each of four components of a result vector.

29. (New) The method as recited in claim 24, wherein, in the programmable mode, the graphics floating point data is operated upon utilizing a predetermined instruction set.

30. (New) The method as recited in claim 29, wherein, in the programmable mode, instructions of the predetermined instruction set are executed per a fragment program.

31. (New) The method as recited in claim 24, wherein the determining is performed utilizing a command associated with the graphics application program interface.

32. (New) The method as recited in claim 31, wherein the command is called by a program that governs operation of the graphics pipeline via the graphics application program interface.